计算机科学 1999Vol. 26№. 9

嵌入式系统的协同设计*)

Embedded System Codesign

郭晓东 刘积仁
(东北大学软件中心 沈阳 110006)

Abstract Embedded system is a heterogeneous system which consists of hardware components and software components. Today the use of ASIC technology and many computer-aided design tools such as high level synthesis tool have created new opportunities for embedded system codesign. This paper emphasizes some important codesign technology such as specification, hardware/software partitioning, synthesis, co-simulation.

Keywords Codesign, Partitioning, Synthesis, Co-simulation

1. 引言

嵌入式系统是由软件和硬件两部分组成的专用控制系统。除了微控制器外，嵌入式系统还包括专用集成电路(ASIC)、现场可编程门阵列(FPGA)、数字信号处理器(DSP)等。它广泛应用于消费类电子产品、通讯系统、车辆控制、遥感等领域，在生产、生活中发挥着越来越重要的作用。

目前嵌入式系统已不再是传统意义的单片机系统。随着32位微控制器、嵌入式实时操作系统的出现，以及VLSI技术的不断进步，嵌入式系统软件及硬件的复杂程度不断提高。同时一些新技术的出现使软件与硬件的界限不再明显，例如，随着高层综合等电子设计自动化工具的不断完善及专用集成电路的广泛应用，复杂的算法可以很容易用硬件的方式来实现，同样RISC技术可以使处理器的部分硬件功能用软件来完成。这些新技术及计算机辅助工具的广泛应用不仅为嵌入式系统的设计注入了新的活力，同时也对嵌入式系统的开发提出了新的课题，传统的开发方法已不能适应新形势的需要。

在嵌入式系统的传统开发方法中，开发人员首先根据以往的开发经验将系统所要实现的功能分解成软件功能及硬件功能两部分。尔后，软件开发人员、硬件开发人员根据分解后的功能分别独立进行开发。当嵌入式软件及嵌入式硬件开发结束后，嵌入式软件下载到可编程单元进行调试。

利用这种方法开发嵌入式系统，其质量及开发进度很难保证，因为在调试阶段，系统的实现可能不满足最初功能规格说明中的某些性能指标，需要重新调整软件与硬件所实现的功能。功能调整可能导致硬件部分的重新设计，致使硬件开发费用提高及系统整体设计周期延长。

随着新技术的不断涌现，为满足系统复杂性及市场竞争的要求，需在缩短开发周期的同时提高系统的开发质量。因此在嵌入式系统的开发过程中，嵌入式软件与硬件的开发不应是独立的、分离的，而应是协同的、一致的。即在软件及硬件实现之前，对软件与硬件所实现的功能进行折衷，以便产生一个最佳的软件、硬件分解方案来满足速度、面积、存储容量、功耗、实时性等一系列技术指标要求。同时在硬件开发(fabrication)之前，应对包含软件、硬件的嵌入式系统所实现的功能进行验证，以确保嵌入式系统的实现与最初的系统功能规格说明相一致。这种软件、硬件协同设计技术不仅可以提高嵌入式系统的开发质量、缩短其开发周期，同时该技术在诸如处理器指令集设计、DSP系统设计、软件加速等领域都有着广泛的应用[1]。

*) 本文受国家863项目资助。郭晓东 博士生，研究兴趣为嵌入式系统的协同综合、协同模拟。刘积仁 教授，博士生导师，研究兴趣为分布式多媒体、嵌入式技术、组件技术等。

2. 嵌入式系统协同设计的几个关键技术

90年代以来，随着高层综合技术的不断完善[2]，软件、硬件协同设计技术受到了学术界的普遍关注。世界各地的研究机构在协同设计技术的各个分支领域进行了广泛的研究，取得了一系列的研究成果，提出了很多针对嵌入式系统协同设计的解决方案，同时构造了与其相应的支撑环境，如CODES[3]、CHINOOK[4]、VULCAN[5]、COSYMA[6]、PTOLEMY[7]、ADEPT[8]。

这一节主要讨论与嵌入式系统协同设计相关的几个关键技术，图1是一个广泛应用于软件、硬件协同设计的框架结构。它由系统功能规格说明，软件/硬件功能分解，软件综合、硬件综合、接口综合，及系统集成等几部分组成。

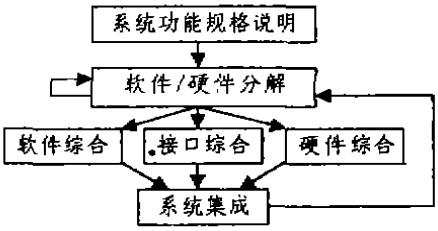

图1 嵌入式系统协同设计流程图

2.1 系统的规格说明与中间表示格式

在系统功能规格说明阶段，首先应给出系统所要实现功能的规格描述。此描述应与系统功能的具体实现无关，同时这种系统功能概念级的描述最好是可执行的，以便对系统功能进行模拟验证。有的支撑环境利用VHDL作为系统算法级的描述语言，有的通过对C语言进行必要的修改与扩充来描述系统行为级的特性[5,6]。

在系统功能分解之前，嵌入式系统的功能描述首先要翻译成有利于分解的统一的、一致的中间表示格式(unified representation)。这种中间表示格式在分解过程中起着重要的作用。利用它，系统软件、硬件功能的折衷及系统功能在软件与硬件之间的转移可以很方便地进行。基于图(graph-based)的表示方法，如CDFG、Petri网[9]，FSM、并发进程等均可作为系统统一的中间表示格式。

2.2 软件/硬件的分解

在嵌入式软件、硬件分解过程中，功能分解算法根据一定的约束条件决定系统中各部份功能的实现方式。软件、硬件分解算法可根据分解的起始点、分解的粒度、分解的自动化程度等不同特性进行分类。一般来说，约束条件下的分解问题是一个NP问题，因此合理选择常用的启发式算法，如贪心算法、模拟退火算法等可以获得较为满意的效果。在分解算法中分解粒度的选择是一个比较关键的问题。分解粒度是指分解算法所作用的基本功能单元的大小。在粗粒度的分解算法中，分解算法选择任务，功能函数作为基本的分解单元，而在细粒度的分解算法中，经常采用功能块、基本的语句作为分解的基本单元。各种分解方案可由费用函数来评估、费用函数的选择对系统性能起着至关重要的作用，采用不同的费用函数可以获得性能迥异的解决方案。

2.3 软件/硬件的综合技术

在系统功能分解之后，组成嵌入式系统的各个构件(component)的实现可利用相应的综合工具，通过对分解后的功能描述的综合处理来获得。硬件构件的综合可由硬件高层综合、逻辑综合、版图综合(layout synthesis)等几个不同阶段构成。软件构件的综合包括软件高级综合、编译、汇编等几个阶段。接口综合包括为硬件构件添加锁存器、FIFOs、地址译码器、为软件构件添加输入/输出操作、同步操作。

在硬件构件综合技术中低层次的逻辑综合技术及版图综合技术比较成熟，而硬件高层综合技术的研究相对起步较晚。它起源于60年代，最初的研究工作仅限于科研机构与大学。但在近十几年中，硬件高层综合技术的研究非常活跃，取得了长足进展，很多研究成果已投入使用。目前在业界，专用集成电路等硬件设备的开发已逐渐从RTL(Register Transfer Level)级设计转化为行为级设计，开发人员无须关注硬件RTL级的具体实现，只需给出系统行为级的功能描述，这一切都得益于硬件高层综合技术的发展。

嵌入式系统的软件综合相对于硬件综合是一个比较新的问题，这是由于传统意义下的软件综合尚不成熟。但是由于嵌入式软件综合与传统意义下的软件综合相比有更多的限制，所以在综合过程中可引入更多的领域知识，因此软件综合技术在嵌入式系统的协同设计中有着很好的应用前景。

2.4 验证技术

在硬件构件开发之前，通过综合获得的软件、硬件实现所组成的系统，其功能的正确性需在系统集成环境中进行验证。验证的方法一般分为形式化验证及模拟验证两种。

形式化验证方法通过运用数学手段来检验系统设计形式化描述的正确性，可分为规格说明验证及实现验证两种。规格说明验证是检验系统的高层模型是否具有某种抽象性质，例如，检验由相互通讯的有限自动机网络来模拟的协议是否死锁；而实现验证是检验通过综合获得的较为低级的系统模型是否与系统的高级模型保持一致，同时检验系统的低级模型是否满足实现级的某些要求。例如，检验一个硬件电路的实现是否与其有限自动机模型相一致，或者检验一个给定的数据流网络实现是否满足实际要求的数据吞吐率。

虽然形式化验证在近些年来取得了很大进步，有着广泛的应用前景，但是模拟验证技术[10]仍然是当前占主导地位的验证技术。嵌入式系统的模拟验证是利用模拟的方法检验由软件、硬件组成的混合系统的正确性。在软件、硬件的协同模拟中有两方面的问题值得注意，一是嵌入式软件的模拟执行速度与其实际的执行速度不同，二是软件、硬件的协同模拟需保持与实际运行中一致的同步操作。协同模拟一般通过以下几种方法来实现。

· 利用全功能模型执行目标代码的方法。首先利用 HDL(Hardware Description Language)编写微处理器或微控制器的全功能模型及相关外部硬件环境的功能模型，然后将嵌入式软件编译成目标码，装入微处理器模拟器的内存模块模拟执行。这种方法只对比较简单的模拟验证有效，或者处理器结构比较简单，或者嵌入式软件代码量比较小。

· 基于指令级模拟器的虚拟系统集成(VSI)。这种方法利用指令级模拟器及硬件逻辑模拟器构成一个 VSI 环境。这个模拟环境提供目标处理器内部状态的寄存器级信息。指令级模拟器的接口将产生精确的总线操作，并通过它来实现嵌入式软件与外围硬件环境的信息交汇。为了实现这种软件及硬件的协同模拟，关键在于处理好指令级模拟器与硬件逻辑模拟器之间的关系，同步和优化是保证正确性及提高执行速度的两个关键技术。

· 利用功能函数模拟硬件环境的方法。当一个新项目开始时，最终执行软件程序的目标系统可能并不存在，为调试嵌入式软件，程序可以交叉汇编到一个已经存在的系统上，硬件的功能可用模型化的函数功能来模拟。这种方法的主要问题是它只提供硬件操作的近似仿真。

如果需要更高的模拟精度与模拟速度，则需采用硬件仿真的方法来满足设计的需要。

3. 实例研究

COSYMA 是由德国 Braunschweig 大学开发的协同综合系统，图 2 是该系统的框架结构，该系统通过引用专用硬件协处理器的方法来提高软件的执行速度。首先利用 C* 语言来描述系统功能的规格说明，C* 语言是在 C 语言的基础上增加了并发描述功能，C* 编译器将系统模型编译成 CDFG 的中间表示格式，分解算法找出计算的瓶颈并利用硬件协处理器处理相应的功能。分解算法分为两层循环，内层循

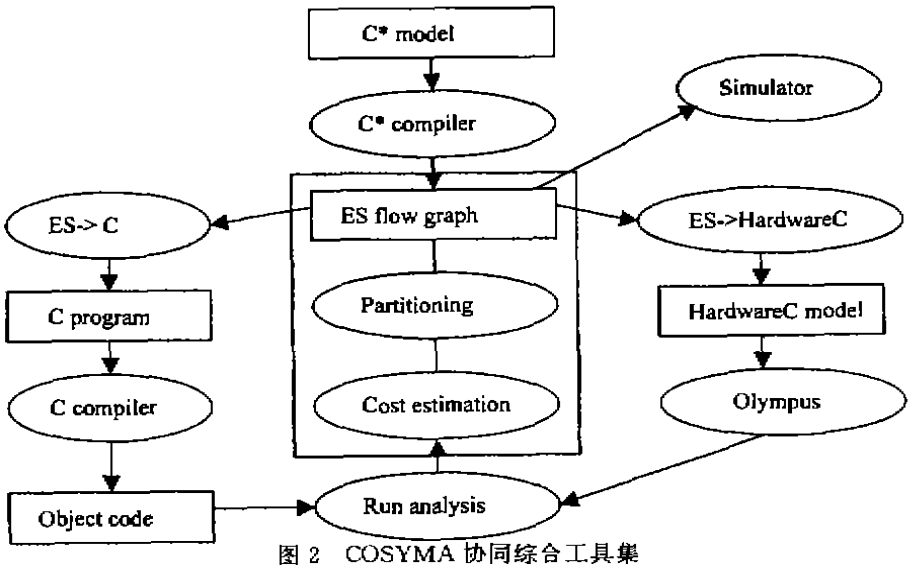

图 2 COSYMA 协同综合工具集

(下转第 19 页)

能界定要有一个明晰的认识：人做人该做的，机器做机器能做的。如果知识工程师尚不能确保所采集的知识对某一智能行为是充分的；且它们的算法是可行的，那么这一智能行为就不能交给计算机去做，例如，对于象"创新"这样通常有赖于灵感爆发的智能，我们自己尚无法勾勒出其发生的来龙去脉，让计算机去做不就勉为其难了吗？对于这样的问题，计算机采用"随机组合枚举"算法也许比"创新"算法来得现实。换句话说，缩小和明确智能系统的应用领域范围，并将其放在辅助而不是替代人类专家的位置上是目前可行的办法。当然，即使这样我们面对的仍然可能是一个巨量的知识库，这个知识库的组织问题也是非常重要的。

从长远来看，对人工智能的基础学科——认知科学的研究才是问题的出路。例如，从发展心理学[6]来看，人的认知发展过程是连续的，任何一个智能行为的发生都不是孤立的，不是可分离的，不能割断发展的前后联系性，就某一单个智能行为就事论事地进行模拟是很片面的，在人工智能问题上也应该有历史观。通过对关于 Nature（自然）和 Nurture（训练）问题的讨论，我们可以获得许多关于智能系统的模型构造设计和学习样本设计的极有价值的启发[7,8]，所以应该给智能系统设计学习子系统，用一种发展的、进化的、循序渐进的观点来实现系统功能的进步[4]。

参 考 文 献

1 姚天顺、等．自然语言理解．北京：清华大学、广西科学技术出版社，1995

2 Witold L. Non-Monotonic Reasoning. Ells Horwood Limited，1990、Introduction

3 Ethrington D W. Reasoning with incomplete information，Pitman Lodon，1988，Introduction

4 危辉．智能脑机制的认知结构核心之元态——表象式语义网及其构造：[北京航空航天大学研究生院博士学位论文]．1998

5 王元元．计算机科学中的逻辑学．北京：科学出版社、1989

6 Collins W A，et al. Developmental Psychology. New York：Macmillan Publishing Company，1991

7 危辉，何新贵．基于颞下皮层结构的视知觉不变性模型．中国智能自动化学会 98 年会．1997

8 危辉、何新贵．视皮层超柱结构的自组织网络模型．第五届全国人工智能联合会议．1998

（上接第 12 页）

环用模拟退火算法初步估计系统的性能，外层循环在 RTL 级进一步修正此值。

参 考 文 献

1 Wolf W H. Hardware-Software Co-Design of Embedded Systems. Proc. of IEEE，1994，82(7)，967～989

2 Mcfarland M C，et al. The High-Level Synthesis of Digital Systems. Proc. of IEEE，1990，78(2)：301～318

3 Buchenrieder K，Veith C. CODES：A Practical Concurrent Design Environment. In：Proc. of the Intl. Workshop on Hardware-Software Co-Design. October 1992

4 Chou P H，et al. The CHINOOK Hardware-Software Co-Synthesis System. In：Proc. Of ISSS. Sept 1995

5 Gupta R K，Micheli G D. Hardware-Software Cosynthesis for Digital Systems. IEEE Design and Test of Computers，1993，10(3)：29～41

6 Ernst R，et al. Hardware-Software Co-Synthesis for Microcontrollers. IEEE Design and Test of Computers，1993，10(4)：64～75

7 Kalavade A，Lee E A. A Hardware-Software Codesign Methodology for DSP Application. IEEE Design and Test of Computers，1993，10(3)：16～27

8 Kumar S，et al. ADEPT：An Unified System Level Modeling Design Environment. In：Proc. of the First Annual RASSP Conference. 114～123

9 Murata T. Petri Nets：Properties，Analysis，and Applications. Proc. of IEEE，1989，77(4)：541～550

10 郭晓东，陈定君，余克清，刘积仁．嵌入式系统设计的模型及方法学．计算机科学，1998，25(5)